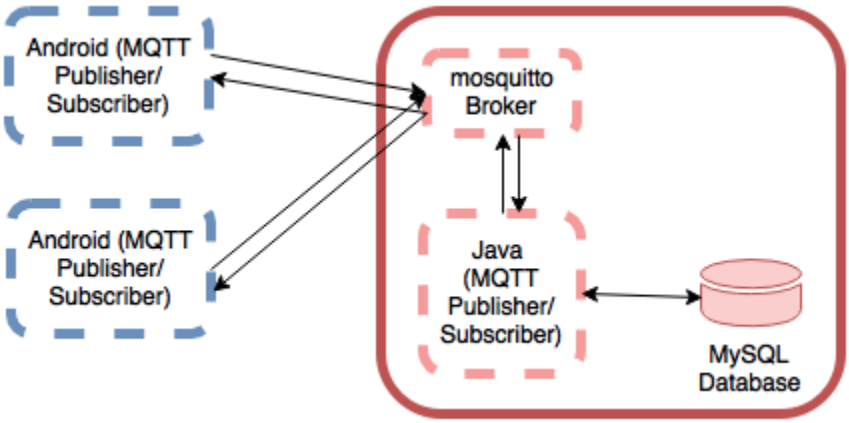
Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Ανάπτυξη Λογισμικού για Δίκτυα και Τηλεπικοινωνίες
Χειμερινό εξάμηνο 2016 - 2017

Το Automation System (Pub/Sub extension) που καλείστε να υλοποιήσετε είναι η εντός σύνδεσης επέκταση της αρχικής εφαρμογής (εκτός σύνδεσης) για την αποφυγή σύγκρουσης μεταξύ δύο τερματικών κινητών συσκευών (Android).

Η αρχιτεκτονική υψηλού επιπέδου του συστήματος φαίνεται στο παρακάτω σχήμα:



Δεύτερο Παραδοτέο

Η **εφαρμογή Android** θα λειτουργεί με δύο καταστάσεις λειτουργίας (offline και online) και θα μεταβάλλει την κατάσταση με βάση την κατάσταση συνδεσιμότητας του κινητού τερματικού (η εναλλαγή θα μπορεί να γίνεται είτε αυτόματα είτε με τη χρήση κουμπιού). Κατά την αυτόματη αλλαγή θα αλλάζει αυτόματα η κατάσταση του κουμπιού καθώς και θα εμφανίζεται κατάλληλο μήνυμα “toast”. Το κουμπί αυτό θα πρέπει να είναι λειτουργικό μόνο στην περίπτωση που υπάρχει συνδεσιμότητα στο τερματικό ώστε να είναι εφικτό να αλλάξει την κατάσταση λειτουργίας της εφαρμογής σε “εκτός σύνδεσης”.

- Σε κατάσταση λειτουργίας offline η εφαρμογή θα παραμείνει ίδια με το πρώτο παραδοτέο (δε θα ληφθούν υπόψιν πιθανές βελτιώσεις του πρώτου παραδοτέου).

- Σε κατάσταση λειτουργίας online, η εφαρμογή χρειάζεται να επεκταθεί κατάλληλα ώστε να αποστέλλει σε πραγματικό χρόνο (publish) και κατάλληλο topic το όνομα του τερματικού, τον τύπο των αισθητήρων, την τρέχουσα μέτρηση των αισθητήρων, την τρέχουσα ημερομηνία και ώρα καθώς και την τρέχουσα τοποθεσία GPS υπό τη μορφή συντεταγμένων σε κεντρικό σύστημα MQTT broker.

  Στις ρυθμίσεις της εφαρμογής χρειάζεται να προστεθούν και οι παράμετροι που είναι απαραίτητες για τη σύνδεση με τον MQTT broker (πχ. IP, Port). Οι ρυθμίσεις των αισθητήρων στην Android εφαρμογή δεν θα λαμβάνονται υπόψιν στην online λειτουργία.

  Ταυτόχρονα θα πρέπει να εγγράφεται επιτυχώς στον MQTT broker σε κατάλληλο topic για την λήψη ειδοποιήσεων σύγκρουσης (subscribe). Τέλος θα πρέπει να λαμβάνει επιτυχώς τα κατάλληλα μηνύματα ειδοποίησης σύγκρουσης (receive message) καθώς και να αναπαράγει ήχο και μήνυμα ειδοποίησης όπως είχε ζητηθεί στο πρώτο παραδοτέο. Σημειώνεται ότι το μήνυμα ειδοποίησης πιθανής σύγκρουσης σε offline λειτουργία θα πρέπει να είναι διαφορετικό από το αντίστοιχο μήνυμα σε online λειτουργία. Επίσης σε κατάσταση online λειτουργίας το μήνυμα θα πρέπει να είναι διαφορετικό για απλή πιθανότητα σύγκρουσης και διαφορετικό για επιβεβαιωμένη πιθανότητα σύγκρουσης.

  Το όνομα της συσκευής θα παράγεται μια φορά, κατά την πρώτη εκτέλεση της εφαρμογής. Η μοναδικότητα του ονόματος μπορεί να επιτευχθεί με τη χρήση της MAC Address του κινητού ή με παραγωγή μοναδικού αναγνωριστικού (UUID).

Ο **MQTT broker** θα βασίζεται στην υλοποίηση του mosquitto και θα εκτελείται τοπικά στο ίδιο σύστημα με την Java εφαρμογή και τη MySQL βάση δεδομένων.

Η **Java εφαρμογή** θα λαμβάνει τις τιμές από από τις δύο εφαρμογές Android (subscribe) σε κατάλληλο topic. Θα υπολογίζει την πιθανότητα σύγκρουσης με βάση τις τιμές των αισθητήρων και τα ορισμένα από τη Java εφαρμογή κατώφλια (περιγραφή παρακάτω). Για κάθε πιθανότητα σύγκρουσης που εντοπίζεται θα αποθηκεύει σε βάση δεδομένων MySQL εγγραφή που θα περιλαμβάνει τα πεδία που έχουν αποσταλεί από το τερματικό και αφορούν την πιθανή σύγκρουση (όνομα του τερματικού, τοποθεσία πιθανής σύγκρουσης υπό τη μορφή συντεταγμένων, τύπο και μέτρηση του αισθητήρα που οδήγησε στον εντοπισμό, ημερομηνία και ώρα ανίχνευσης).

Στη συνέχεια θα αποστέλλει μήνυμα ειδοποίησης σύγκρουσης (publish) στον MQTT broker με το κατάλληλο topic, απ' τον οποίο θα ειδοποιείται αυτόματα μόνο το εμπλεκόμενο με τη σύγκρουση τερματικό.

Σε περίπτωση που εντοπιστούν δυο πιθανές συγκρούσεις (μια για κάθε τερματικό Android) με χρονική απόσταση όχι μεγαλύτερη του ενός δευτερολέπτου θα θεωρείται επιβεβαιωμένη πιθανότητα σύγκρουσης μεταξύ των δύο τερματικών. Θα μεταβάλλει τις αντίστοιχες δύο εγγραφές της βάσης ως “επιβεβαιωμένες” και θα αποστέλλει κατάλληλο μήνυμα ειδοποίησης σύγκρουσης (publish) στον MQTT broker με το κατάλληλο topic, απ’ τον οποίο θα ειδοποιούνται αυτόματα και τα 2 Android τερματικά.

Η Java εφαρμογή θα συνοδεύεται από γραφικό περιβάλλον παρακολούθησης των πιθανών συγκρούσεων βασισμένο σε Java FX. Συγκεκριμένα, θα περιλαμβάνει 2 οθόνες που θα παρουσιάζονται με τη μορφή tab:

- Η πρώτη οθόνη θα περιλαμβάνει πολυκριτηριακή αναζήτηση με βάση οποιοδήποτε πεδίο της βάσης, ή συνδυασμό αυτών. Η εφαρμογή θα επιστρέφει τα αποτελέσματα της αναζήτησης από τη βάση δεδομένων και θα τα παρουσιάζει σε μορφή λίστας στο γραφικό περιβάλλον. Οι εγγραφές που αντιστοιχούν σε επιβεβαιωμένη πιθανότητα σύγκρουσης θα παρουσιάζονται με διαφορετικό χρώμα παρασκηνίου από τις μη επιβεβαιωμένες. Η ανανέωση της λίστας καθώς και η δυνατότητα νέας αλλαγής θα πραγματοποιείται με τη χρήση των αντίστοιχων κουμπιών. Σε περίπτωση υπερμεγέθους λίστας θα υποστηρίζεται σελιδοποίηση.

- Η δεύτερη οθόνη θα περιλαμβάνει τη ρύθμιση των κατωφλιών των αισθητήρων (με αντίστοιχο τρόπο με αυτό που είχε υλοποιηθεί στις ρυθμίσεις του Android τερματικού στα πλαίσια του πρώτου παραδοτέου).

Τέλος είναι υποχρεωτική η χρήση διαφορετικών threads στην Java εφαρμογή για την εξυπηρέτηση της λειτουργίας των διαφορετικών tasks (γραφικά, εκτέλεση αλγορίθμων, λήψη μηνυμάτων, αποστολή μηνυμάτων, επικοινωνία με τη βάση δεδομένων κλπ).

Τεχνολογίες συστήματος ανάπτυξης:

1. Java Oracle SE 8 (Java FX included)
2. Android API 4.1 ή νεότερο
3. Android Studio (Εργαλείο ανάπτυξης της Android εφαρμογής)
4. IntelliJ IDEA (Εργαλείο ανάπτυξης της Java εφαρμογής)
4. MQTT mosquitto broker
5. Eclipse Paho (Java & Android MQTT client)
6. MySQL Community server
7. JDBC

Κατά τη διαδικασία ανάπτυξης της εργασίας είναι υποχρεωτική η χρήση του εργαλείου Git (Version Control) και της πλατφόρμας gitlab (anapgit.scanlab.gr) που παραχωρείται για τις ανάγκες του μαθήματος.

Η υλοποίηση της εφαρμογής θα πρέπει:

1. Να υπακούει στις αρχές του αντικειμενοστραφούς προγραμματισμού
2. Να γίνεται σωστή και αποδοτική οργάνωση του κώδικα σε κλάσεις και πακέτα.
3. Να είναι όσο το δυνατό παραμετροποιήσιμη και δυναμική γίνεται.
4. Να γίνεται σωστή και αποδοτική διαχείριση της μνήμης.

Προαιρετική επέκταση με Google Vision API (bonus 20%)

Περί Google Vision API:

Το Google Vision API επιτρέπει στους προγραμματιστές να ζητήσουν την ανάλυση μιας φωτογραφίας/εικόνας μέσω ενός REST interface, με το οποίο αποστέλλεται η εικόνα μαζί με άλλες πληροφορίες και επιστρέφεται απάντηση - ανάλυση σε μορφή JSON.

Υπάρχουν πολλοί διαθέσιμοι τύποι ανάλυσης εικόνας, όπως FACE_DETECTION, LANDMARK_DETECTION, LOGO_DETECTION, LABEL_DETECTION κλπ.
Στο πλαίσιο της εργασίας θεωρείται περισσότερο χρήσιμο το LABEL_DETECTION, το οποίο “χαρακτηρίζει” αυτά που βλέπει. Ωστόσο, μπορούν να συμπεριληφθούν διαφορετικοί τύποι ανάλυσης.

Google Vision API στην ανίχνευση σύγκρουσης:

Η επέκταση που καλείστε να υλοποιήσετε (προαιρετικά, με bonus 20% στο συνολικό βαθμό) θα συμπεριλαμβάνει τη χρήση της κάμερας των κινητών τερματικών Android, καθώς και το Google Vision API.

Στην περίπτωση που ανιχνεύεται οποιαδήποτε επικείμενη σύγκρουση, αυτά θα φωτογραφίζουν και θα αναλύουν το περιβάλλον τους, ώστε να παράξουν αντίστοιχη ειδοποίηση. Οι ειδοποιήσεις μπορούν είτε να αποτελούνται από προκαθορισμένους ήχους για κάποια είδη αντικειμένων (πχ. κολόνα, αυτοκίνητο, άνθρωπος, κλπ), είτε να παράγονται με text-to-speech ανάλογα με την ανίχνευση που έγινε, το οποίο και αυξάνει την ποικιλία των ειδοποιήσεων.

Εφόσον υλοποιήσετε το bonus τμήμα της εργασίας, χρειάζεται να υπάρχει και κάποιο κουμπί απενεργοποίησης αυτής της λειτουργίας στην Android εφαρμογή, ώστε να συνεχίζει απρόσκοπτα η λειτουργία της και χωρίς αυτό.

Χρήσιμοι σύνδεσμοι για το Google Vision API:

Getting started with Google Cloud Vision
[example] Image Detection Using Android Device Photos
Google Vision API Java classes

Επισήμανση: Για τη χρήση του Google Vision API απαιτείται πιστωτική κάρτα. Ωστόσο παρέχεται δωρεάν δοκιμαστική χρήση για 2 μήνες.